



LAWRENCE  
LIVERMORE  
NATIONAL  
LABORATORY

# DOs and DON'Ts in Developing In-House Industrial Hygiene Software

C. Chen, T. Lowe

May 11, 2006

American Industrial Hygiene Conference and Exposition  
Chicago, IL, United States  
May 13, 2006 through May 18, 2006

## **Disclaimer**

---

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.



---

# ***DOs and DON'Ts in Developing In-House Industrial Hygiene Software***

**C. Chen, T. Lowe**  
**Lawrence Livermore National Laboratory**  
**Livermore, CA**



This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

UCRL-CONF-221315



# *Introduction*



- **Is NOT!:**
  - **Justification to design in-house or to procure commercial software product**
  - **Justification for project resources**
  - **Justification for Return on Investment**
  - **The only process for software application design**



# *Introduction (cont.)*



- **Assumes:**
  - **Management buy-in**
  - **Justification and decision has been made to design in-house software application**
  - **Adequate resources are available for software project lifecycle**



# *Introduction (cont.)*



- 
- **Historical Background**
  - **Project Initiation**
  - **Project Issues**
  - **Software Implementation**
  - **Software Tools**
  - **Lessons Learned – DOs and DON'Ts**
  - **Summary**



# *Historical Background*



- **STS – Sample Tracking System**
- **First attempt to integrate multiple spreadsheets, databases and paper records to electronic records**
- **Records stored in multiple locations and formats**
- **Central file server, early 1990's technology**



# *Historical Background (cont.)*



- **STAR Phase I – Sample Tracking And Reports**
- **To improve on the shortfalls of STS system**
  - **Add more data fields to the data entry screens**
  - **Add functionality to meet changing business needs**



# *Historical Background (cont.)*



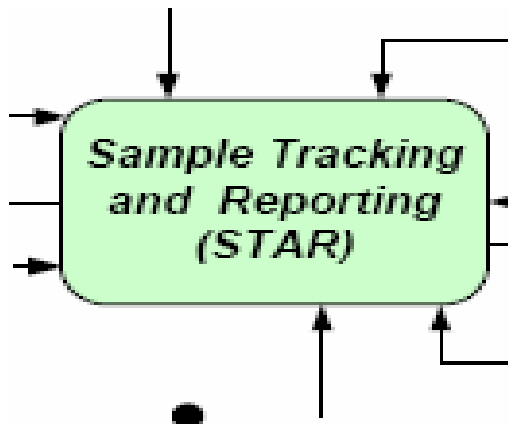
- **To improve on the shortfalls of STS System (cont.)**
  - **Improve data entry integrity using validated lists**
  - **Add audit trail function to track entry and changes**
  - **Add system and data security**



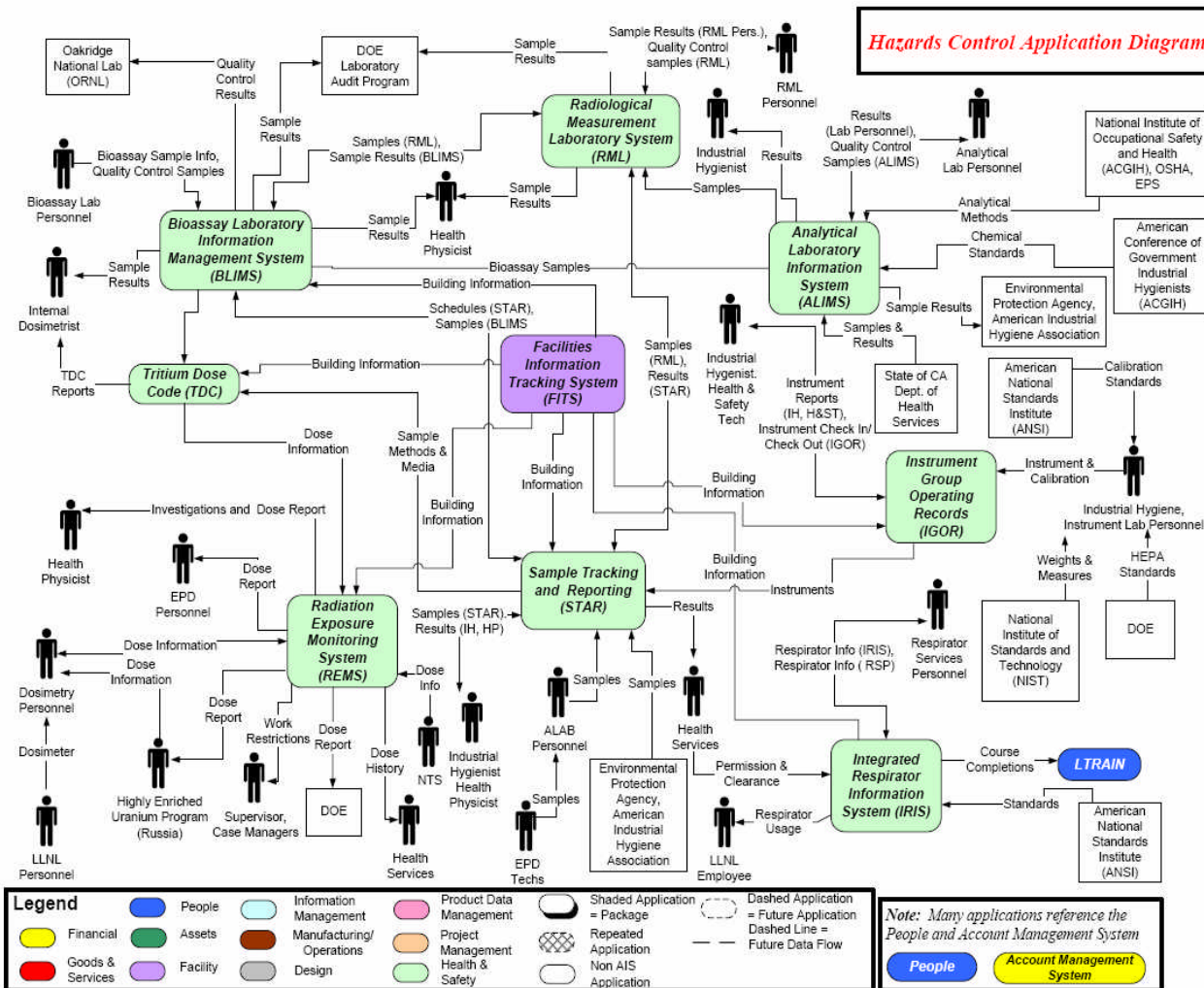
# *Historical Background (cont.)*



- Original project was a “stand alone” application that morphed into an integrated network of applications



# Historical Background (cont.)



# *Project Initiation*



- **Facilitated sessions where all stakeholders provided input in 8-hour sessions**
- **Consensus-driven, all the stakeholders got most of their needs met**
- **After 2 sessions, all “requirements” were gathered**
- **Programmers went away and created end product to meet requirements with very little interaction or feedback**



# *Project Issues*



- **Institutional culture**
  - **No bottom line for \$\$\$ with dedicated programming staff versus job shop**
  - **Data owners did not want to distribute data electronically**
  - **Business requirements driven by audit findings**
  - **Did not have full management buy-in**



# *Project Issues (cont.)*



- **End user issues**
  - **Multiple management chains**
    - **No single authority for decision making or path for conflict resolution**
    - **No accountability**
  - **Minimum computer literacy level not defined**
  - **Availability of resources not consistently allocated**
    - **PCs vs Mac vs terminal servers (software, hardware and operating systems)**
    - **Personnel for participation in project development**



# *Project Issues (cont.)*



- Requirements often not realistic and/or measurable
  - Ex. “Easy to use”, “Intuitive”
- Led to different levels of expectations
- All requirements had same priority ranking
  - “must have”, “should have” or “nice to have” were ranked with the same priority of importance
  - Majority of requirements were “nice to have”
    - “fantasy” because these were NEVER used by requesters
    - Technology was not available



# *Project Issues (cont.)*



- **Stakeholders required data entry screens to use the “One size fits all” approach**
  - **Industrial Hygienist, Health Physicists, Health and Safety Technicians use the same screens**
  - **Various sampling types were entered using the same data entry screen**
  - **Strong resistance to have multiple screens with specific functional needs**
  - **Functionality requirements of each stakeholder were mutually exclusive**
    - **Ex. - one data field required by one user would not function for any other user**



# *Project Issues (cont.)*



- **Lack of ownership of the application resulted in finger pointing when application had problems**
- **Data entry screens based on existing paper/manual formats**
  - **Inefficient use of screen capability**
  - **Users were accustomed to “scratch pad”, line-out and write-in any changes in paper margin**



# *Project Issues (cont.)*



- **No requirements specified by stakeholders for data output**
  - **Part of culture was to enter “nice to have” data**
  - **Database was considered data storage**
  - **No thought was given to meaningful data retrieval**
  - **Data for reporting was hand-entered into personal spreadsheets, “This is my data and not for anyone else to see”**



# *Software Implementation*



- **Hardware limitations**
  - Due to high costs, desktop computers were not widely available
  - PCs, Macs and VAX/VMS terminals using central server technology
  - Developed to lowest common denominator (vt220 monitor)
- **Software limitations**
  - Terminal emulators for multiple OS platforms with multiple versions for each OS



# *Software Implementation (cont.)*



- **Affects on return on investment for hardware and software**
  - **Higher costs to support multiple hardware and software platforms**
    - **Need more support personnel with expertise**
  - **High costs for programmers to create a specific data entry screen for each platform and OS version**
    - **System migrations cannot be accomplished for legacy systems with unsupported hardware and software**
  - **Large resources required to test for each platform/OS version for a given screen**
    - **Often for a single user**



# *Software Implementation (cont.)*



- **Lack of management buy-in**
  - **Testing had low participation**
    - **Participating testers got their needs**
    - **Users who opted not to test were most unhappy when software released for production user**
  - **Training was offered**
    - **When optional, few participated in training**
    - **When training was mandatory, complaints, selective comprehension usually had same result as optional training**



# *Software Implementation (cont.)*



- **Software documentation**
  - Documentation provided
  - Rarely used, most user “wing it”
  - Not enough resources to keep documentation current
- **External data input and output**
  - Analytical laboratories
  - Result reports
  - “paper form” model does not address electronic data processing, “free type” data entry not optimal for queries



# *Software Development Tools*



- **CASE Tools (Computer Aided Software Engineering)**
  - Provided easy mechanism for multiple developers to keep same “look and feel” for each screen
  - Repository for documenting programming parameters for consistency



# ***Lessons Learned – DOs and DON'Ts***



- **DON'T try to please everybody**
  - **Do use ranking system for requirements**
- **DO use K.I.S.S. philosophy**
- **DO have project manager**
  - **Knowledgeable about business requirements**
  - **To enable executive decisions when there are conflicting issues**
  - **Has trust of senior management and credibility with end users**
- **DO use formal project management methodology**
  - **Ex. IEEE**



# *Lessons Learned – DOs and DON'Ts*

## *(cont.)*



- **DO consider external data input and output**
  - **DON'T design a “garbage in, garbage out” program**
  - **DO design database to provide structure for meaningful generic data retrieval (Ex. - reports)**
- **DO have minimum standards**
  - **DON'T have multiple hardware platforms**
  - **DO have common software**
    - **Browsers**
    - **Operating systems**
    - **Versions**
  - **DO have minimum computer literacy levels for end users**



# ***Lessons Learned – DOs and DON'Ts (cont.)***



- **DO have formal testing plan**
  - **DON'T release without documentation that acceptance criteria was met**
  - **DO follow standards for testing and documentation (Ex. IEEE, NQA-1, etc)**
- **DO have a training program in place**
  - **DON'T allow access to application to users that have not completed training program**
  - **DO keep training material updated**



# ***Lessons Learned – DOs and DON'Ts (cont.)***



- **DO apply Software Quality Assurance principles**
- **DO use formal documentation (Ex. IEEE standards for guidance)**
  - **DO document application requirements (data input, output usage and data sharing), also minimize “change orders”**
  - **DON'T perform maintenance requests without documentation**
  - **DO document “bug” fixes**
  - **DO perform verification and validation**



# *Conclusion*



- **Old issues with development never go away, just return in different form**
  - **What is state of the art today is tomorrow's legacy**
  - **Address hardware and software issues**
  - **Provide adequate support personnel**
  - **Ensure management support level**



# *Conclusion (cont.)*



- **Keep an open mind, anticipate issues**
  - **New applications merge with current applications**
  - **New business requirements may require modification of current software or migration to new system**
- **Remember and apply lessons learned**
  - **Don't “reinvent the wheel”**



# *Questions*



---

**Chuck Chen**

**925-422-8098**

**chen2@llnl.gov**

**Tim Lowe**

**925-422-8430**

**tlowe@llnl.gov**

